Proceedings of the 2001 IEEE/RSJ
International Conference on Intelligent Robots and Systems
Maui, Hawaii, USA, Oct. 29 – Nov. 03, 2001

A Voronoi-Based Hybrid Motion Planner

Mark Foskey Maxim Garber Ming C. Lin Dinesh Manocha
Department of Computer Science
University of North Carolina at Chapel Hill
http://www.cs.unc.edu/~geom/voronoi/vplan

Abstract

We present a hybrid path planning algorithm for rigid and articulated bodies translating and rotating in a 3D workspace. Our approach generates a Voronoi roadmap in the workspace and combines it with "bridges" computed by a randomized path planner with Voronoi-biased sampling. The Voronoi roadmap is computed from a discrete approximation to the generalized Voronoi diagram (GVD) of the workspace, which is generated using graphics hardware. By this use of the GVD, portions of the path can be generated without random sampling, substantially reducing the number of random samples needed for the full query. The planner has been implemented and tested on a number of benchmarks. Some preliminary comparisons with a randomized motion planner indicate that our planner performs more than an order of magnitude faster in several challenging scenarios.

1 Introduction

The problem of automated motion planning has seen important progress in the past decade. The *probabilistic roadmap* (PRM) approach has been of particular importance, providing a straightforward method for handling configuration spaces of high dimension with good efficiency [13]. In this paper we use ideas from earlier geometric or "criticality based" methods to improve the performance of a PRM planner on certain classes of problems.

In PRM planning, a graph characterizing the topology of the free space is built up by generating robot configurations at random and, among those for which the robot is not in collision, attempting to connect nearby configurations using some very simple, fast planner. The method is probabilistically complete, in the sense that PRM planners can be made arbitrarily likely to find any solution by allowing a sufficient running time. They perform well for a wide variety of problems, but they can be slow when the robot must pass through a narrow passage to reach the goal.

The earlier criticality based methods rely on an explicit geometric description of the configuration space to provide a data structure that can be searched for a path. Criticality based algorithms are typically complete: They either return a correct path or an indication that none exists. In low dimensions, many workable algorithms of this type have been implemented [16]. For arbitrary dimensions, known algorithms are prohibitively difficult to implement and have complexity exponential in the dimension of the C-space. An example of a general algorithm is the roadmap planner based on Whitney's stratified sets [1]. For our purposes, one benefit of criticality based methods is that they are not hindered by narrow passages.

Main Results: We present a hybrid path planning algorithm for free-flying rigid and articulated bodies translating and rotating in a 3D workspace. Our approach utilizes a global geometric analysis of the workspace to generate an approximate path in configuration space. We then identify invalid segments of this estimated path, for which the robot is in collision, and compute linking subpaths or "bridges" [3] to replace the invalid segments. These bridges are generated by a randomized planner with carefully restricted sampling.

Our global geometric analysis uses a discrete approximation of the generalized Voronoi diagram (GVD) of the workspace, computed using graphics hardware [9]. The key distinctive features of our approach are:

- We generate an estimated path based on the Voronoi diagram of the scene, and only use randomized planning for parts of that path.
- When we use randomized planning, it is guided by information in the Voronoi diagram.

Our planner has been implemented and applied to a number of benchmarks. We have also compared its performance with a one-shot planner in the general PRM family developed by Hsu et al. at Stanford [12]. Some preliminary results indicate that our hybrid planner for rigid bodies is more than an order of magnitude faster than the PRM planner on many of these benchmarks.

The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we introduce terminology related to the Voronoi diagram, and in Section 4 we explain our algorithm. In Section 5 we give implementation details and present performance results. In Section 6 we analyze the performance of the algorithm, and in the final section we conclude and indicate areas of future work.

2 Related Work

2.1 Voronoi Diagrams in Motion Planning

Generalized Voronoi diagrams have long been used as a basis for motion planning algorithms [4, 5, 17, 23]. The GVD represents the connectivity of a space but has a dimension lower by one, and (in three dimensions) it is composed of surfaces of maximal clearance.

0-7803-6612-3/01/$10.00©2001 IEEE

The disadvantage of using the GVD has always been that it is difficult to compute robustly and efficiently. Recently, several approaches to this problem have been proposed. Vleugels and Overmars [22] give an algorithm that applies spatial subdivision and isosurface extraction techniques to acquire an approximate model of the diagram. Choset and Burdick [4, 5] define a related structure called the *hierarchical generalized Voronoi graph*, which they compute using continuation methods. More recently, Wilmarth, Amato, and Stiller [23, 24] have shown how points on the GVD can be found without computing a representation of the entire set. Finally, Hoff et al. [9] have introduced a method that uses graphics hardware to generate a discrete model at a specified resolution. We use this latter method in our work.

The Voronoi diagram can be used in several ways:

1. The Voronoi diagram of the configuration space (or the Voronoi graph, described below) may simply be searched for a path, once the start and finish points have been linked to the diagram [2, 4, 5, 17]. Such an approach is only practical if the number of degrees of freedom (dof) is fairly low, say three or less.
2. A path found using the GVD of the workspace may be used to provide intermediate points to serve as temporary attractive wells for a potential field planner [10].
3. The GVD of the workspace can be used to bias sample generation in a randomized planner [8, 11, 20, 23, 24].

Our planner uses method 1 above to generate its initial workspace path. It also uses method 3 when computing linking subpaths, because sampling is indirectly biased by the GVD.

2.2 Randomized Planning

The PRM method [14] was developed independently by Kavraki and Latombe [13] and Overmars and Švestka [18, 19]. A PRM planner generates samples at random in configuration space, attempting to connect each sample by a simple C-space path to one of the points already found. Over time, the graph thus produced will tend to represent the connectivity of the C-space reasonably well, and a query can be rapidly performed by linking the search points to the graph and then searching the graph. Many variations on this idea have been developed. We use the one-shot planner of Hsu et al. [12] as a baseline for comparison and as a subroutine in our work. Some details of its algorithm are given in Section 4.4.

3 Background and Notation

Generalized Voronoi Diagram: Let a set of geometric objects, or *sites*, be denoted $s_1, s_2, \ldots, s_n$. For each site s_i, define a distance function $d_i(\mathbf{x}) = \text{dist}(s_i, \mathbf{x})$. The *Voronoi region of* s_i is the set $V_i = \{\mathbf{x} \mid d_i(\mathbf{x}) \leq d_j(\mathbf{x}) \, \forall j \neq i\}$.

The collection of regions $V_1, \ldots, V_n$ is called the *generalized Voronoi diagram* or *GVD*, which partitions the space into cells suitable for proximity queries. The (ordinary) Voronoi diagram corresponds to the case when each s_i is an individual point.

The boundaries of the regions V_i are called *Voronoi boundaries*, which are loci of points equidistant to at least two sites. Sometimes we use the term GVD to refer to the union of the Voronoi boundaries, rather than the collection of Voronoi cells.

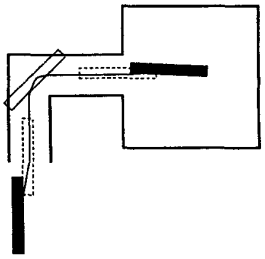

Figure 1: An estimated path for the robot. The solid rectangles indicate the initial and goal configurations. The robot is in collision with the environment as it turns around the corner.

In three dimensions, the intersection of two Voronoi regions is a *Voronoi face*, the intersection of multiple Voronoi faces is a *Voronoi edge*, and the intersection of multiple Voronoi edges is a *Voronoi vertex*. Together the Voronoi edges and vertices form a graph, the *generalized Voronoi graph* (GVG). The GVD is a *deformation retract* of the workspace [2] which ensures that it reflects the topology of the workspace. The GVG, by contrast, can be disconnected even if the workspace is connected [5].

4 Algorithm

Our hybrid planning algorithm can be outlined as follows.

1. Compute the generalized Voronoi graph.
2. Search the GVG to find a path for a point robot in the workspace.
3. Use shape analysis to choose orientations for the actual robot along the point robot path, generating an approximate path in C-space.
4. Find all portions of the estimated path for which the robot is colliding with the obstacles. See Figure 1 for an example.
5. Use Voronoi-biased randomized planning to replace each path segment where the robot is colliding with the environment.

Details are given in the rest of the section.

4.1 Computing the GVG

Our method for computing the generalized Voronoi diagram is based on the algorithm presented by Hoff et al. [9], and it uses standard polygon rasterization hardware. We compute the discrete Voronoi diagram in slices. For each slice L_z determined by a given z value, and each site s_i, there is a real-valued distance function d_i^z given by $d_i^z(x, y) = d_i(x, y, z)$. In words, the value of $d_i^z(x, y)$ is the distance in 3-space from (x, y, z) to s_i. As an example, consider the case of a Voronoi site s_i that is the single point $(0, 0, 1)$, and the slice L_0. In this case, $d_i^0(x, y) = \sqrt{x^2 + y^2 + 1}$.

The graph of d_i^z is a surface for which we generate a triangular approximation called a *distance mesh*. We assign each site s_i a unique identifying color, and we render its distance mesh d_i^z in that color by using a parallel projection. After all the distance meshes are rendered, we have, for each pixel, the identity of the

nearest site, determined by the color, and the distance to that site recorded in the depth buffer. The algorithm reads back the color buffer and the depth buffer. The depth buffer contains the distance field, i.e. the distance to one of the nearest obstacle for each pixel in the slice.

In our discrete representation of the GVD, we regard the Voronoi boundaries as lying between neighboring pixels. A discrete Voronoi edge consists of a sequence of *pixel edges*, with each pixel edge bounded by two pixels. The endpoints of pixel edges are *pixel vertices*. If each pixel is regarded as filling a small solid cube, then the pixel edges and vertices are the edges and vertices of the cube. We compute the GVG by scanning the 3-D pixel map, two slices at a time, seeking pixel edges whose neighboring pixels exhibit at least three different colors. We store the resulting locations in an edge list representation. The vertex data structure contains the coordinates of the point and the clearance distance to the obstacles (because it is a Voronoi vertex, the point will be equidistant to at least four sites). The edge data structure has a list of sampled coordinates of points on the edge, and the minimum clearance distance for the whole edge. Note that all the sample points are restricted to a uniform grid, so that the vertex and edge points do not lie on the actual Voronoi boundaries, but instead on nearby grid points.

As we construct the diagram, we construct a list for each Voronoi site recording all the Voronoi vertices to which that site is a nearest neighbor. When a Voronoi vertex is found, its nearest neighbors are determined by the colors in the adjacent pixels, and it is added to the lists for those sites.

4.2 Generating a Path in the Workspace

After generating the GVG, we use it to find an approximate path in the workspace for the robot to follow, called the *workspace path*. Define a *query configuration* to be an initial or goal configuration, and a *query location* to be the projection to $\mathbf{R}^3$ of a query configuration. Then the workspace path links the initial and goal query locations.

Before we can search the GVG for a path, we need to link the query locations to the GVG. To link a query location, we first determine the Voronoi cell containing it. We then compute line segments from the query location to each Voronoi vertex of the cell, and eliminate any segments that pass through obstacles.

We add the query locations and their linking line segments to the GVG data structure as (formal) Voronoi vertices and edges. Each newly added edge contains a list of points and the value of the minimum distance to the environment.

After linking the query locations to the GVG, we use a generalized single-source shortest paths algorithm, where the length of a path is determined by a combination of the Manhattan distance along the path and the maximal clearance over the whole path. This path, the workspace path, is a solution to the query for a point robot, and it satisfies a partial criterion of maximal clearance.

It is possible for the GVG to be disconnected even when the workspace is connected. It would be possible to add additional edges to the graph to ensure proper connectivity, using information derived from the full discrete Voronoi diagram. The resulting graph would be similar to the HGVG of Choset and Burdick [5]. However, for simplicity our planner simply uses the PRM algorithm on the query locations when no workspace path is found using the GVG. In practice we have found such cases to be rare.

4.3 Orienting the Robot

After finding the workspace path, we must choose an orientation for the robot at each point on the path. To do this, we determine a major axis for the robot, and align it with the tangent vector of the path, as determined by a finite difference estimate. In this respect, or method bears comparison with methods for "stick" or "ladder" robots, such as that described in Choset, Mirtich, and Burdick [6].

For a complex shape, there are many reasonable definitions of the "major axis." For our purposes, we want an axis around which the robot fits as tightly as possible. To determine such an axis, we use linear regression to compute a best-fit line approximating the vertices of the robot. This line is chosen to minimize the root mean square of the (Euclidean) distances of the vertices to the line. The origin of the robot is defined to be the center of gravity of the vertices. For articulated robots, we define a standard pose, and compute the major axis with respect to that pose.

Once we have determined how to align the specified major axis of the robot, it is still free to rotate about that axis. The choice of orientation about the major axis (i.e., the "roll") is made arbitrarily. We simply make sure that, up to discrete approximation, the orientation varies continuously as the robot traverses the path.

4.4 Bridging Invalid Segments

In this section, we explain how the estimated path is modified into a final path for the robot. First, using a simple straight-line local planner, we attempt to connect each configuration with its successor. Configurations for which the robot is colliding with the obstacles, or which cannot be connected to a neighbor, are marked "invalid".

The path has now been decomposed into valid segments, for which the robot is free, alternating with invalid segments, for which it is not. For each invalid segment, we apply the randomized planner [12], with the start and goal given by the free configurations immediately before and after the invalid segment. This planner maintains trees of free configurations rooted at the start and finish. At each iteration (called an *expansion iteration*), it chooses a configuration p from one of the trees, generates new configurations in a neighborhood of p, and retains those which can be linked to p by a free path. The local planner terminates when the two trees are connected. This algorithm automatically biases sampling towards configurations known to be free.

The configuration space for the planner is defined to be the tightest axis-aligned box that contains bounding balls for the robot at both query locations (see Figure 2). The only degrees of freedom that are restricted in a special way for the randomized planning phase are the translational coordinates. The orientation of the robot as a whole, as well as joint positions for an articulated robot, are given full freedom.

It is possible for the robot to get into a tight spot for which the restricted configuration space does not provide enough room for the robot to maneuver from the beginning of the invalid segment to the end of it. To handle such situations we use a simple expe-

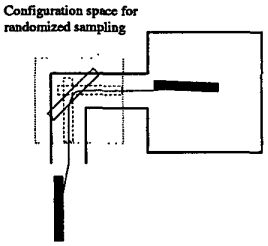

Figure 2: The two dashed rectangles indicate valid configurations that will be linked by randomized planning. The larger dotted rectangle indicates the restricted C-space used.

dient: If, after a fixed number of expansion iterations, the planner has not linked the two ends of the invalid segment, the planner's configuration space is enlarged to the full original C-space, and randomized planning is resumed. If this is not successful after another predetermined number of iterations, then it is assumed that the heuristics guiding the initial path estimate have failed, and the planner simply uses the randomized planner to link the original start to the original finish. Such cases tend to be difficult for the randomized planner as well, so the time spent in trying to use the estimated path (from GVDs) is typically a small fraction of the total time.

4.5 Localized Sampling

While an invalid segment is being bridged, what was a narrow passage on the scale of the entire scene is now a relatively open area within the restricted configuration space (see Figure 2). However, there may be a portion of the invalid segment which constitutes a narrow passage even on this smaller scale. To increase sampling in these bottleneck areas, we generate a new configuration near the narrowest point on the invalid segment ("narrowest" being measured in terms of distance from Voronoi sites, i.e., the obstacles). We find the new configuration by uniform random sampling in a neighborhood of the narrowest point.

We then perform two randomized sampling steps, one linking the beginning of the invalid segment to the new configuration near the narrowest point, and the other linking the new configuration to the end of the invalid segment. The new configuration acts as a seed, causing a number of configurations to be generated near the narrowest point on the narrow passage.

If the Voronoi site distance at the narrowest point is greater than half the radius of the robot's bounding ball, then this operation is not performed since in practice we have found that a single randomized planning step works well in such cases.

5 Implementation and Performance

The algorithm has been implemented in C++. We used PQP [15] for collision detection during randomized planning, and Magic Software written by D. Eberly to compute the major axis of the robot.

For rigid robots, we used several benchmark scenarios, described below. We compare computation times using our algorithm with the best times we were able to achieve using the randomized planner alone.

Duct: Two open areas separated by a channel with two right angle turns. The robot is a narrow box. See Figure 3.

Walls: A series of six walls, four of which have small holes through which the robot, a narrow box, must pass (Figure 4). We require the robot to pass from one end of the maze to the other, through all four holes. This benchmark was designed at Texas A & M university [21].

Piano: Eight Chairs, a table, and a piano (Figure 5). The goal is to move the piano through the window. The window is smaller than the convex hull of the piano, forcing the piano to rotate in order to reach the goal. This benchmark was provided by LAAS, Toulouse.

2D Maze: A maze with a spike-shaped robot (Figure 6).

3D Maze: A stack of four connected mazes, each similar to 2D Maze shown in Figure 6.

Pegs: Tilted pegs that a human figure must avoid (Figure 7).

The results of the benchmarks are summarized in Table 5. The *Voronoi resolution* is the resolution of the discrete Voronoi diagram, defined as D/S, where D is the length of the main diagonal of the scene, and S is the distance between sampled distance values. All times are in seconds on a 300 MHz MIPS R12000 processor.

The randomized planner has several adjustable parameters that can affect performance. The timings for the PRM planner reflect our best efforts to choose parameters that give optimal performance of the PRM planner for the given scenes.

For articulated robots, we have tested the performance of our framework on a number of benchmarks. These include:

Crane: A CAD model of a crane complex composed of more than 128,000 triangles (Figure 8). The model contains 143 separate polyhedral parts, which are rendered in distinct colors in the figure. The robot is an articulated robot arm part with 10 degrees of freedom. The model was provided by LAAS.

Maze: The maze is the one shown in Figure 6. The robot is a series of boxes with 9 degrees of freedom.

The timing results (without any optimization) of these benchmarks are summarized in Table 5.

6 Analysis and Discussion

In this section, we discuss the performance of our planner. We consider properties of the robot and workspace that affect the performance.

If there is a path for the robot which is, to the precision of our Voronoi computation, wider than the bounding ball for the robot, then our planner will generally find it very rapidly, without randomized planning. Thus, for our planner, any region of the workspace wider than the bounding ball of the robot does not correspond to a narrow passage in C-space. We therefore define a *workspace narrow passage* to be a portion of the GVG for which the site distance is less than the radius of the robot's bounding ball. By the above observations, any C-space narrow passage corresponds to some part of a workspace narrow passage.

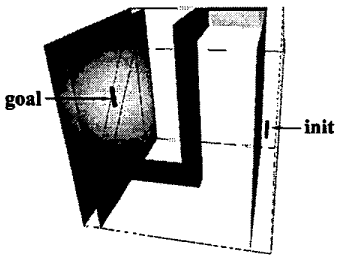

Figure 3: Duct: An elongated, box-shaped robot must traverse a bent duct to reach the goal configuration, shown behind the partially transparent wall.

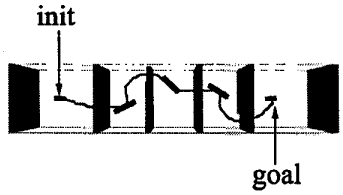

Figure 4: Walls: A narrow box-shaped robot must pass through the square openings in four different walls. The path shown is the solution found by our planner.

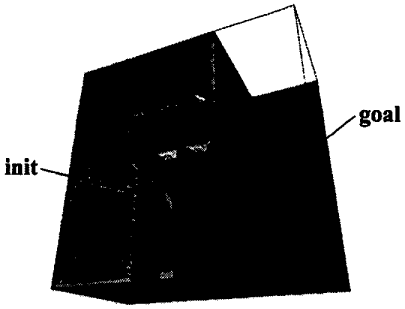

Figure 5: Piano: The piano must avoid the other furniture and maneuver through the window to reach the goal configuration in the next room.

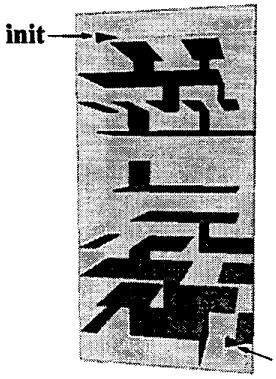

Figure 6: 2D Maze: The wedge shaped robot must navigate through the maze of walls and passages to reach its goal at the bottom right corner of the map.

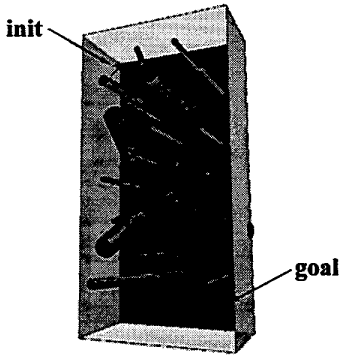

Figure 7: Pegs: Twenty-five pegs of different sizes, tilted at various angles. It is navigated by the stick figure robot shown.

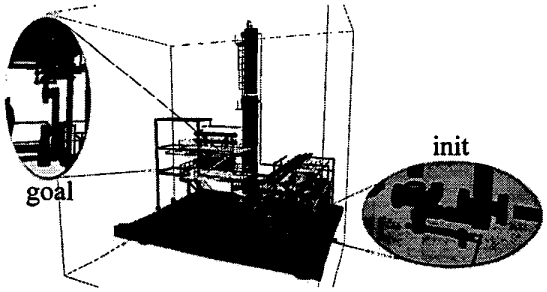

Figure 8: Crane: A complex crane assembly.

Scene	Res	GVG	Query	PRM	Gain
Duct	128	10.4	0.70	894	88
Walls	128	2.98	0.55	20.3	5
Piano	64	5.05	41.9	603	14
2D Maze	128	2.14	5.74	341	43
3D Maze	128	24.9	23.8	450	9
Pegs	64	2.87	19.7	97.7	4

Table 1: Benchmark timings. **Res:** Voronoi resolution. **GVG:** Voronoi graph computation time. **Query:** query phase, after Voronoi computation. **PRM:** the randomized planner alone. **Gain:** speedup factor.

Scene	Res	GVG	Query
Crane	64	138.65	334.71
Maze	128	5.73	59.66

Table 2: Benchmark timings in seconds for articulated robots. **Res:** Voronoi resolution. **GVG:** Voronoi graph computation. **Query:** query phase, after Voronoi computation.

Because invalid segments are determined by collision of the robot with the environment, they can only occur in workspace narrow passages, or along a segment joining a query configuration to the GVG. Thus our planner primarily uses randomized planning in a subset of the workspace narrow passages. It may seem paradoxical that narrow passages, which can be notoriously difficult for randomized planners, are precisely where we use randomized planning. This strategy in fact works well because of the three techniques we use to bias sampling:

- We initially restrict randomized planning to a region delimited by the endpoints of the invalid segment, increasing the chances that we will generate more samples in the narrow passages. Essentially, in the context of this restricted C-space, the narrow passage becomes a relatively open area.
- When a passage is especially narrow, measured in terms of the workspace, we seed the PRM planner with an additional configuration near the narrowest point. These additional seed configurations have the effect of intensifying sampling in the most restricted areas.
- We use a PRM planner that generates new samples near samples already found at both ends of the narrow passage. Trees of configurations rooted at the two ends of the narrow passage have a high probability of growing into it. This phenomenon is discussed in terms of *expansive components* in [12].

7 Conclusion

We have introduced a hybrid planner that uses simplified global geometric analysis to generate an estimated path, and then uses randomized planning guided by the generalized Voronoi diagram, to modify the estimated path into a collision free path. We have tested the planner on several benchmarks and found that it can be over an order of magnitude faster than a comparable purely randomized planner.

Acknowledgements

We would like to thank Kenny Hoff for providing us his Voronoi computation software. We are also grateful to David Hsu and Jean-Claude Latombe for helpful discussion on PRM and providing us with an earlier version of their randomized planner. Finally, we thank Jean-Paul Laumond and Nicola Simeon for for providing us the piano and crane models This research was supported in part by ARO Contract DAAG55-98-1-0322, DOE ASCII Grant, NSF grants ACI-9876914, DMI-9900157 and NSF IIS-982167, ONR Young Investigator Award, and Intel.

References

[1] J. Canny. *The Complexity of Robot Motion Planning*. ACM – MIT Press Doctoral Dissertation Award Series. MIT Press, Cambridge, MA, 1987.

[2] J. F. Canny and B. Donald. Simplified voronoi diagrams. *Discrete and Computational Geometry*, 3:219–236, 1988.

[3] J. F. Canny and M. C. Lin. An opportunistic global path planner. *Algorithmica*, 10:102–120, 1993.

[4] H. Choset and J. Burdick. Sensor based planning, part ii: Incremental construction of the generalized voronoi graph. *IEEE Conference on Robotics and Automation*, 1995.

[5] H. Choset and J. Burdick. Sensor based planning: The hierarchical generalized voronoi graph. *Workshop on Algorithmic Foundations of Robotics*, 1996.

[6] H. Choset, B. Mirtich, and J. Burdick. Sensor based planning for a planar rod robot: Incremental construction of the planar rod-hgvg. *IEEE Conference on Robotics and Automation*, 1997.

[7] B. R. Donald. Motion planning with six degrees of freedom. Master's thesis, MIT Artificial Intelligence Lab., 1984. AI-TR-791.

[8] L. Guibas, C. Holleman, and L. Kavraki. A probabilistic roadmap planner for flexible objects with a workspace medial-axis-based sampling approach. In *Proc. of IROS*, 1999.

[9] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. *Proceedings of ACM SIGGRAPH 1999*, pages 277–286, 1999.

[10] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Interactive motion planning using hardware accelerated computation of generalized voronoi diagrams. *IEEE Conference on Robotics and Automation*, pages pp. 2931–2937, 2000.

[11] C. Holleman and L. Kavraki. A framework for using the workspace medial axis in PRM planners. *International Journal of Computational Geometry and Applications*, 9((4 & 5)):495–512, 1999.

[12] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications*, 9((4 & 5)):495–512, 1999.

[13] L. Kavraki and J. C. Latombe. Randomized preprocessing of configuration space for fast path planning. *IEEE Conference on Robotics and Automation*, pages 2138–2145, 1994.

[14] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, pages 12(4):566–580, 1996.

[15] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.

[16] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[17] C. Ó'Dúnlaing, Micha Sharir, and C. K. Yap. Retraction: A new approach to motion-planning. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pages 207–220, 1983.

[18] M. H. Overmars. A random approach to motion planning. Report RUU-CS-92-32, Dept. Comput. Sci., Utrecht Univ., Utrecht, Netherlands, October 1992.

[19] M. H. Overmars and P. Švestka. A probabilistic learning approach to motion planning. In *Algorithmic Foundations of Robotics*. A. K. Peters, Wellesley, MA, 1995.

[20] C. Pisula, K. Hoff, M. Lin, and D. Manocha. Randomized path planning for a rigid body based on hardware accelerated voronoi sampling. In *Proc. of 4th International Workshop on Algorithmic Foundations of Robotics*, 2000.

[21] D. Vallejo, C. Jones, and N. Amato. An adaptive framework for single shot motion planning. In *Proc. of IROS*, 2000.

[22] Jules Vleugels and Mark Overmars. Approximating generalized Voronoi diagrams in any dimension. Technical Report UU-CS-1995-14, Department of Computer Science, Utrecht University, 1995.

[23] Steven A. Wilmarth, Nancy M. Amato, and Peter F. Stiller. Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space. *IEEE Conference on Robotics and Automation*, 1999.

[24] Steven A. Wilmarth, Nancy M. Amato, and Peter F. Stiller. Motion planning for a rigid body using random networks on the medial axis of the free space. *Proc. of the 15th Annual ACM Symposium on Computational Geometry (SoCG'99)*, 1999.